
CS 696

Introduction to Grid Computing:
Lecture #14: ZSI

Mary Thomas
San Diego State
Tuesday, 20-Mar-07

What is ZSI?

- Zolaris Soap Interface
- Focus is on parsing/generating SOAP messages
- Intended to be used by other client/server systems:
 - provides limited dispatching capabilities
- 3 primary functional aspects:
 - SOAP Message Handling
 - Client-Server (invocation/dispatch)
 - Wsdl2py: web service development using WSDL

How it Works

- Parses inputstream, generates a DOM tree
 - Processes *header*
 - Processes *body*: creates local Python objects
 - *dispatches* to handler; creates output objects
 - Creates *SoapWriter* object for data, namespace, etc.
 - Serializes messages for client, sends to client
 - Handles faults
- Transport neutral

ZSI: Diverse Uses

- pyGridWare/pyGSI/pyGlobus:
 - http://dev.globus.org/wiki/Python_Core
 - Clarens: <http://clarens.sourceforge.net/>
 - Amazon:
 - <http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=7>
 - Portals: Plone/Zope:
 - <http://plone.org/documentation/how-to/import-soap-client-object>
 - Mod_python (Apacher Web Server)
 - <http://www.modpython.org/>
 - Cheshire search engine:
 - <http://www.cheshire3.org/>
-

ZSI Capabilities

CORE

- Data Conversions
 - Typcode classes:
TC.TypeCode, TC.Any,
TC.SimpleType
 - Basic Message Handling -
ParsedSOAP
 - Serializing Data: SoapWriter
module
 - Fetching remote data:
resolvers module
 - Fault & Utilities modules
- Client/Server:
 - client module
 - dispatch module
 - Web Services
development:
 - WSDL2py
 - Creates set of client (or
server) codes from a
WSDL document

ZSI: Installation

Download from <http://pywebsvcs.sourceforge.net/>

```
[gidget:python/libs/ZSI-2.0] mthomas% python setup.py --help-commands
```

Standard commands:

```
build          build everything needed to install  
build_py        "build" pure Python modules (copy to build directory)  
build_ext       build C/C++ extensions (compile/link to build directory)  
build_clib      build C/C++ libraries used by Python extensions  
build_scripts   "build" scripts (copy and fixup #! line)  
clean           clean up output of 'build' command  
install        install everything from build directory  
install_lib     install all Python modules (extensions and pure Python)  
install_headers install C/C++ header files  
install_scripts install scripts (Python or otherwise)  
install_data    install data files  
sdist           create a source distribution (tarball, zip file, etc.)  
register        register the distribution with the Python package index  
bdist           create a built (binary) distribution  
bdist_dumb      create a "dumb" built distribution  
bdist_rpm       create an RPM distribution  
bdist_wininst   create an executable installer for MS Windows
```

ZSI Installation - Test code

- Since the most useful documentation and How-To examples are the unit test codes, you should study up on unit testing in python.
- Test directories:
 - ZSI-2.0/tests - ZSI TC modules/SOAP
 - ZSI-2.0/tests/wSDL2py - Web service clients based on auto-generated code based on WSDL
 - ZSI-2.0//samples/Echo: client and server example
 - Warning: ***** this example does not work ******

Unit Testing: a word

- Based on Erich Gamma's JUnit & Kent Beck's Smalltalk testing framework.
- Test classes subclass the *TestCase* class of the *unittest* module, part of Python 2.x or greater
- Test-case methods: individual test methods take no parameters, return no values
- *TestSuite*: A test suite is a composite test consisting of a number of *TestCases*.
- No use of try-catch --> if exception, this is a failure: let unittest catch them
- You can set up 'good' and 'bad' test cases

TestCase: frequently used methods

- setUp(): prepares new instance for test-case
- tearDown(): cleans up
- assertEquals(arg1,arg2,msg) returns msg if arg1 != arg2
- assertNotEqual(arg1,arg2,msg) returns msg if arg1 == arg2
- assertRaises(exceptionSpic,callable,*args): returns msg if arg1 != arg2

ZSI Core Capabilities

- Data Conversions
 - Typcode classes: `TC.TypeCode`, `TC.Any`, `TC.SimpleType`
- Basic Message Handling - ParsedSOAP
- Serializing Data: SoapWriter module
- Fetching remote data: resolvers module
- Fault & Utilities modules
- Client/Server: client and dispatch modules

ZSI TypeCode Classes

- Converts data between SOAP and Python
 - ZSI supports Python numeric and string types
- All typecodes have prefix **TC**.
- Serialization control: fine grain
 - pname: parameter name of XML element
 - All typecodes take pname arg
 - Can be a list: (namespace-URI, localname) or string
 - aname: attribute name: values override pname

ZSI DataTypes: ZSI.TC class - pulled from index list:

TypeCode - Parent Class

Any - used for dynamic typing

AnyElement

AnyType

SimpleType

List

ComplexType

Struct, Arrays

Strings:

Base64Binary, Base64String

HexBinaryString

XMLString

Enumeration

URI

Integer

Enumeration, lbyte, llong,...

Floats

decimal

FPfloat, FPdouble

Apache

XML

QName

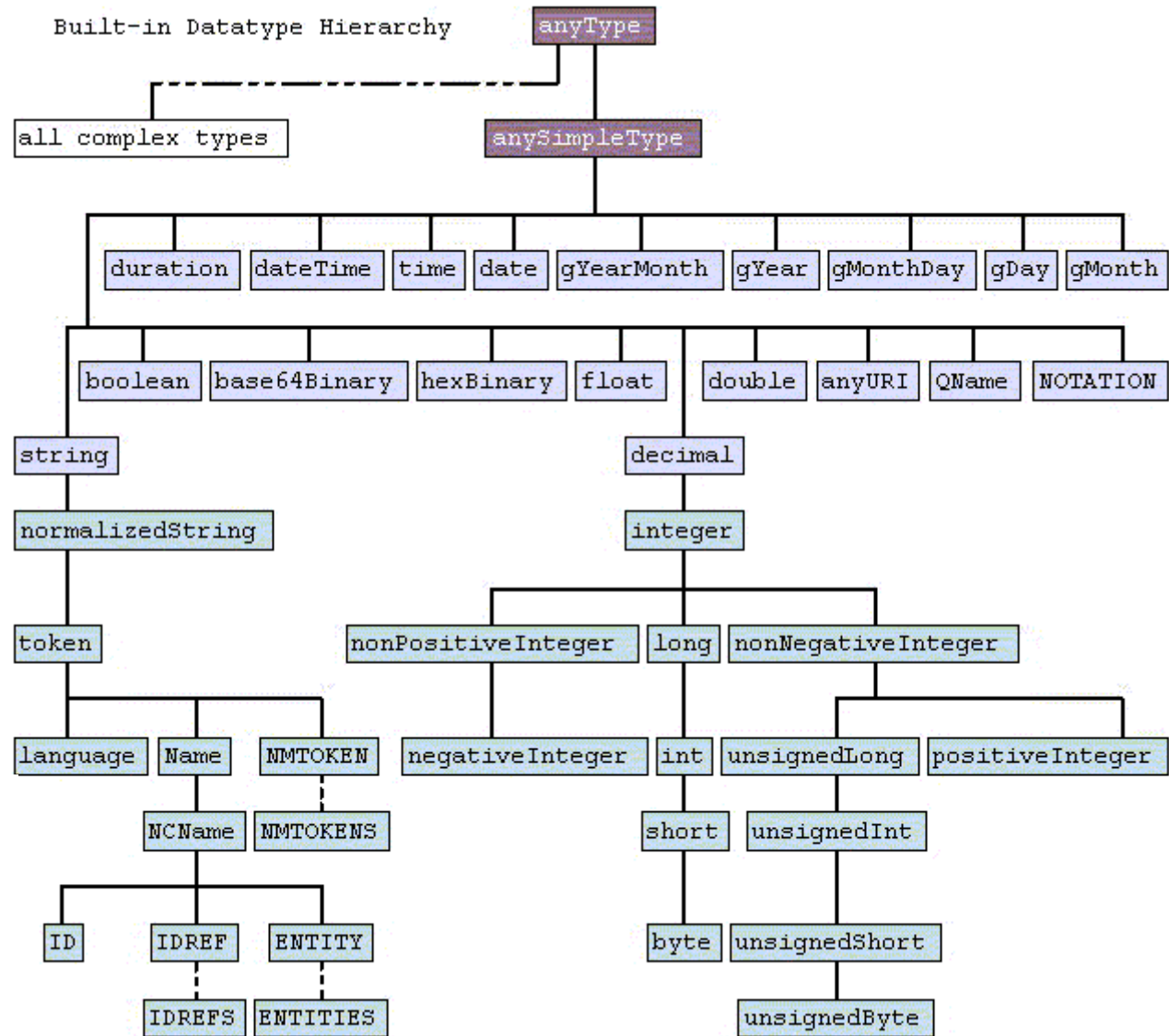
Token

Boolean

Decimal

Union

Maps to SOAP Datatypes



- ur types
- built-in primitive types
- built-in derived types
- complex types

SOAPpy Datatypes: SOAPpy.Types class

| | | | |
|--|---|---|---|
| anyType voidType stringType untypedType IDType NCNameType NameType ENTITYType IDREFType languageType NMTOKENType QNameType tokenType normalizedStringType CDATAType booleanType decimalType floatType doubleType durationType timeDurationType | dateTimeType recurringInstantType timeInstantType timePeriodType timeType dateType gYearMonthType gYearType centuryType yearType gMonthDayType recurringDateType gMonthType monthType gDayType recurringDayType hexBinaryType base64BinaryType base64Type binaryType | anyURIType uriType uriReferenceType NOTATIONType ENTITIESType IDREFSType NMTOKENSType integerType nonPositiveIntegerType non_Positive_IntegerType negativeIntegerType negative_IntegerType longType intType shortType byteType nonNegativeIntegerType non_Negative_IntegerType | unsignedLongType unsignedIntType unsignedShortType unsignedByteType positiveIntegerType positive_IntegerType compoundType structType headerType bodyType arrayType typedArrayType faultType SOAPException RequiredHeaderMismatch MethodNotFound AuthorizationFailed MethodFailed |
|--|---|---|---|

ZSI Core Capabilities

- Data Conversions
 - Typcode classes: TC.TypeCode, TC.Any, TC.SimpleType
- **Basic Message Handling - ParsedSOAP**
- Serializing Data: SoapWriter module
- Fetching remote data: resolvers module
- Fault & Utilities modules
- Client/Server: client and dispatch modules

ZSI.parse: SOAP Message Handling

- ParsedSOAP: convert the text to a DOM tree and parse SOAP elements.
- Instance data:
 - reader -- the DOM reader
 - dom -- the DOM object
 - ns_cache -- dictionary (by id(node)) of namespace dictionaries
 - id_cache -- dictionary (by XML ID attr) of elements
 - envelope -- the node holding the SOAP Envelope
 - header -- the node holding the SOAP Header (or None)
 - body -- the node holding the SOAP Body
 - body_root -- the serialization root in the SOAP Body
 - data_elements -- list of non-root elements in the SOAP Body
 - trailer_elements -- list of elements following the SOAP body

ZSI Core Capabilities

- Data Conversions
 - Typcode classes: TC.TypeCode, TC.Any, TC.SimpleType
- Basic Message Handling - ParsedSOAP
- **Serializing Data: SoapWriter module**
- Fetching remote data: resolvers module
- Fault & Utilities modules
- Client/Server: client and dispatch modules

Serializing Data: SoapWriter module

- SOAP output formatter.
- Instance Data:
 - memo -- memory for id/href
 - envelope -- add Envelope?
 - encodingStyle --
 - header -- add SOAP Header?
 - outputclass -- ElementProxy class.

ZSI Core Capabilities

- Data Conversions
 - Typcode classes: TC.TypeCode, TC.Any, TC.SimpleType
- Basic Message Handling - ParsedSOAP
- Serializing Data: SoapWriter module
- Fetching remote data: resolvers module
- Fault & Utilities modules
- Client/Server: client and dispatch modules

resolvers module: fetch data

- SOAP message parsing
- Class NetworkResolver
 - A resolver that support strings and XML.
- Class MIMEResolver
 - Multi-part MIME resolver -- SOAP attachments.

ZSI Core Capabilities

- Data Conversions
 - Typcode classes: TC.TypeCode, TC.Any, TC.SimpleType
- Basic Message Handling - ParsedSOAP
- Serializing Data: SoapWriter module
- Fetching remote data: resolvers module
- **Fault & Utilities modules**
- Client/Server: client and dispatch modules

Fault & Utilities modules

- Class Fault will process SOAP faults
 - Faultcode, messages, etc.
 - Serialize to SOAP using AsSOAP function
 - Can use with unittest faults as well
- Class ZSIException
 - Subtype of Python Exception class
 - ZSIException
 - ParseException - ParsedSOAP exceptions

ZSI Core Capabilities

- Data Conversions
 - Typcode classes: TC.TypeCode, TC.Any, TC.SimpleType
- Basic Message Handling - ParsedSOAP
- Serializing Data: SoapWriter module
- Fetching remote data: resolvers module
- Fault & Utilities modules
- Client/Server: client and dispatch modules

ZSI: Dispatch (Server) Mechanisms

- Simplest: AsServer
 - Uses python BaseHTTP.... For client server
- AsCGI
- ZSI interface for viewing messages: add this to client connector:
 - `tracefile=sys.stdout`
 - Seems to work best with client.
 - `python -d debug` - provides some information

ZSI.dispatch

- Common Gateway Interface (CGI)
 - standard protocol for interfacing external application software with an information server, commonly a web server.
- Functions:
 - **AsServer**(port=80, modules=None, docstyle=False, nsdict={}, typesmodule=None, rpc=False, addr="")
 - **AsCGI**(nsdict={}, typesmodule=None, rpc=False, modules=None)
 - Dispatch within a CGI script.
 - **AsHandler**(request=None, modules=None, **kw)
 - Dispatch from within ModPython.
 - **AsJonPy**(request=None, modules=None, **kw)
 - Dispatch within a jonpy CGI/FastCGI script.
 - multi-threaded, OO, CGI/FastCGI/mod_python/html-templating

ZSI AsServer server: Server & Client

SERVER

=====

```
#!/usr/bin/python
def hello():
    return "ZSI is Going to Work!"
from ZSI import dispatch
dispatch.AsServer(port=8899, rpc=True)
```

CLIENT

=====

```
import sys, time
from ZSI.client import NamedParamBinding as npb
b=npb(url='http://localhost:8899', tracefile=sys.stdout)
print "hello! trying...\n"
b.hello()
```

ZSI AsServer server: Request/Response

```
[gidget:dev/simplecases/asServer] mthomas% python -d myc.py  
hello! trying...
```

Mon Mar 19 10:51:45 2007 REQUEST:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-  
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ZSI="http://www.zolera.com/schemas/ZSI/"  
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" SOAP-  
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:Header></SOAP-ENV:Header><SOAP-  
ENV:Body><hello></hello></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Mon Mar 19 10:51:45 2007 RESPONSE:

```
200
```

```
OK
```

```
-----
```

```
Server: ZSI/1.1 BaseHTTP/0.3 Python/2.5
```

```
Date: Mon, 19 Mar 2007 17:51:45 GMT
```

```
Content-type: text/xml; charset="utf-8"
```

```
Content-Length: 455
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-  
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ZSI="http://www.zolera.com/schemas/ZSI/"  
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><SOAP-  
ENV:Header></SOAP-ENV:Header>  
<SOAP-ENV:Body>  
<helloResponse id="o58c20" xsi:type="xsd:string">ZSI is Going to Work!  
</helloResponse>  
</SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Example: test_t7.py

```
#!/usr/bin/env python
import unittest, sys
from ZSI import *
class t7TestCase(unittest.TestCase):
    "Test case wrapper for old ZSI t7 test case"
    def checkt7(self):
        ps = ParsedSoap(text)
        tcdict = TC.Apache.Map('c-gensym1')
        tclist = TC.Apache.Map('c-gensym1', aslist=1)
        d = tcdict.parse(ps.body_root, ps)
        self.assertEqual(d, { u'a':123, '\x00\x01':456 })
        print 'as dictionary\n', d
        l = tclist.parse(ps.body_root, ps)
        self.assertEqual(l, [('\x00\x01', 456), (u'a', 123)])
        print '\n', '=' * 30
        print 'as list\n', l
        print '\n', '=' * 30
        sw = SoapWriter()
        sw.serialize(d, tcdict)
        print >>sys.stdout, sw
        print '\n', '=' * 30
        sw = SoapWriter()
        sw.serialize(l, tclist)
        print >>sys.stdout, sw
```

Example: test_t7.py

```
def makeTestSuite():
    suite = unittest.TestSuite()
    suite.addTest(unittest.makeSuite(t7TestCase, "check"))
    return suite
def main():
    unittest.main(defaultTest="makeTestSuite")

text = """
<SOAP-ENV:Envelope xmlns:
SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
  xmlns:xmlsoap="http://xml.apache.org/xml-soap">
<SOAP-ENV:Body>
<c-gensym1 xsi:type="xmlsoap:Map">
  <item>
    <key xsi:type="SOAP-ENC:base64">AAE=</key>
    <value xsi:type="xsd:int">456</value>
  </item>
  <item>
    <key xsi:type="xsd:string">a</key>
    <value xsi:type="xsd:int">123</value>
  </item>
</c-gensym1>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
"""

if __name__ == "__main__": main()
```

Test Codes

- There are several test codes provided in directory “tests”
- They all work fine
- Mostly, they exercise the ZSI classes for type, parsing SOAP.
- There are no working examples for dispatch/client
- Do not use the ones in the sample directory, or the developers guide

ZSI: wsdl2py

Generating clients from WSDL docs

ZSI: wsdl2py

- Uses existing WSDL document to generate client and server applications
 - WSDL doc must be part of the WebService
 - Defines SOAP bindings: rpc, document
- wsdl2py:
 - generates python code for WSDL components
- wsdl2dispatch:
 - run after wsdl2py
 - Generates an interface, usually run with http
- Then, run server.py and client.py codes
- Bindings:
 - RPC - remote procedure calls/messages
 - Document: exchanges of documents

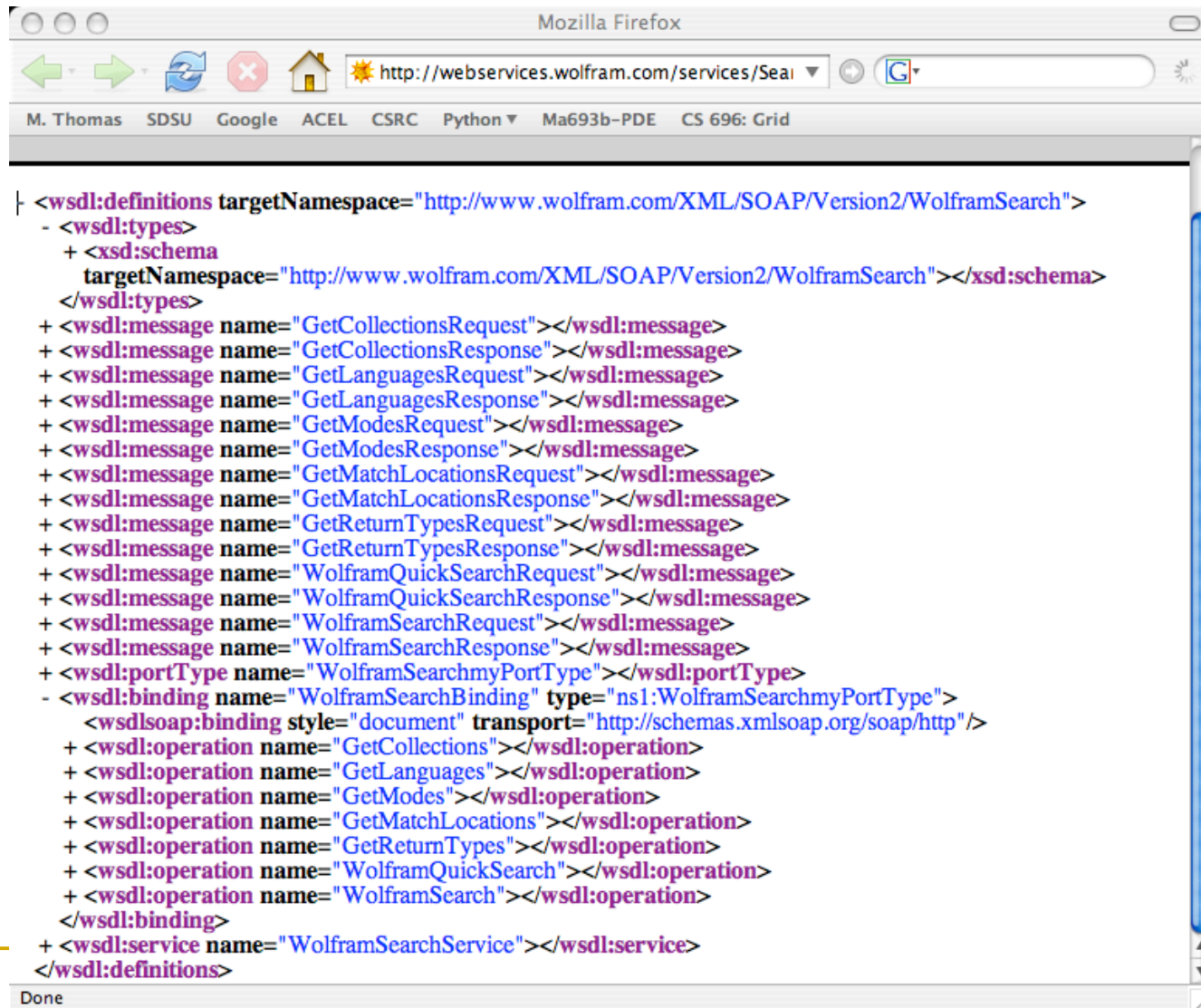
ZSI.wsdl2py

- Generates a client stub module which has
 - ❑ Class CLASSNAMELocator(*kws)
 - ❑ Class Service
 - ❑ Class PortType
 - ❑ Message classes: two for each PortType
- Types module
 - ❑ Maps types in WSDL to ZSI types

Wolfram.com Webservice: Search



http://webservices.wolfram.com/services/SearchServices/WolframSearch2.wsdl



```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://www.wolfram.com/XML/SOAP/Version2/WolframSearch">
  <wsdl:types>
    <xsd:schema targetNamespace="http://www.wolfram.com/XML/SOAP/Version2/WolframSearch"></xsd:schema>
  </wsdl:types>
  <wsdl:message name="GetCollectionsRequest"></wsdl:message>
  <wsdl:message name="GetCollectionsResponse"></wsdl:message>
  <wsdl:message name="GetLanguagesRequest"></wsdl:message>
  <wsdl:message name="GetLanguagesResponse"></wsdl:message>
  <wsdl:message name="GetModesRequest"></wsdl:message>
  <wsdl:message name="GetModesResponse"></wsdl:message>
  <wsdl:message name="GetMatchLocationsRequest"></wsdl:message>
  <wsdl:message name="GetMatchLocationsResponse"></wsdl:message>
  <wsdl:message name="GetReturnTypesRequest"></wsdl:message>
  <wsdl:message name="GetReturnTypesResponse"></wsdl:message>
  <wsdl:message name="WolframQuickSearchRequest"></wsdl:message>
  <wsdl:message name="WolframQuickSearchResponse"></wsdl:message>
  <wsdl:message name="WolframSearchRequest"></wsdl:message>
  <wsdl:message name="WolframSearchResponse"></wsdl:message>
  <wsdl:portType name="WolframSearchmyPortType"></wsdl:portType>
  <wsdl:binding name="WolframSearchBinding" type="ns1:WolframSearchmyPortType">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="GetCollections"></wsdl:operation>
    <wsdl:operation name="GetLanguages"></wsdl:operation>
    <wsdl:operation name="GetModes"></wsdl:operation>
    <wsdl:operation name="GetMatchLocations"></wsdl:operation>
    <wsdl:operation name="GetReturnTypes"></wsdl:operation>
    <wsdl:operation name="WolframQuickSearch"></wsdl:operation>
    <wsdl:operation name="WolframSearch"></wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="WolframSearchService"></wsdl:service>
</wsdl:definitions>
```

Run wsdl2py on the Wolfram WSDL document

Running the command:

```
mthomas% wsdl2py -bu http://webservices.wolfram.com/services/SearchServices/WolframSearch2.wsdl
```

Caused the following files to be created:

```
[gidget:test/wsdl2py/wolf] mthomas% ls
```

```
total 88
```

```
drwxr-xr-x  6 mthomas mthomas  204 Feb 27 17:32 .
drwxr-xr-x 40 mthomas mthomas 1360 Feb 27 17:22 ..
-rwxr-xr-x  1 mthomas mthomas 1163 Feb 27 17:22 WolframSearchService.py
-rw-r--r--  1 mthomas mthomas 5086 Feb 27 17:32 WolframSearchService_services.py
-rw-r--r--  1 mthomas mthomas 24972 Feb 27 17:32 WolframSearchService_services_types.py
-rwxr-xr-x  1 mthomas mthomas  2589 Feb 27 17:22 test_WolframSearch.py
```

Service file: contains portType,
locator, message classes

types file: schema
declarations and instances

Uses the name attribute of the wsdl:server element to name files:

```
<wsdl:binding name="WolframSearchBinding" type="ns1:WolframSearchmyPortType">
<wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
```

WolframServiceSearch.py

```
import sys, time
from ZSI.ServiceContainer import AsServer
from WolframSearchService_services_server import WolframSearchService
class Service(WolframSearchService):
def soap_WolframSearch(self, ps):
    rsp = WolframSearchService.soap_WolframSearch(self, ps)
    msg = self.request
    t1 = time.time()
    opts = msg.Options
    rsp.Result = result = rsp.new_Result()
    if opts.Query == 'Newton':
        result.TotalMatches = 1
        result.Matches = match = result.new_Matches()
        item = match.new_Item()
        item.Title = "Fig Newtons"
        item.Score = 10
        item.URL = 'http://www.nabiscoworld.com/newtons/'
        match.Item = [item]
    result.SearchTime = time.time() - t1
    return rsp
if __name__ == "__main__" :
    port = int(sys.argv[1])
    AsServer(port, (Service('test'),))
```

wSDL2py tests

- Test directory:
 - Tests/wSDL2py
 - All use class: EchoServer_services_types.py
 - Most work, but not all